第一章：

一、名词解释

1. 操作系统：操作系统是管理计算机硬件与软件资源，为应用程序提供共同的公共服务的系统软件

4. 实时操作系统：指系统对特定输入做出反应的速度，并以控制发出实时信号的对象的一种操作系统。即及时响应外部事件请求，在规定时间内完成对该事件的处理，并控制所有实时任务使他们协调一致地运行的一种操作系统。

5. 互斥共享：系统中的资源在某一时间内只允许一个进程访问

三、填空题：

1. 硬件系统　软件系统
3. 硬件资源　软件资源
5. 共享性　异步性

第二章：

一、名词解释：

3. 临界区：在多个进程或线程中访问共享资源的代码段，需互斥控制保证同时只有一个进程可访问临界区。

4. 进程同步：指多个进程中发生的事件存在着某种时序关系，必须协同动作，相互配合，以共同完成一个任务。

三、填空题：

1. 动态性　异步性
5. 动态　静态
6. 间接制约
8. 管道通信　消息队列　共享内存
9. 就绪状态　执行状态　阻塞状态

五.问答题

2. S_1:无前驱 S_2:S_1 S_3:S_1 S_4:S_2 S_5:S_2 S_6:S_3
S_7:S_4;S_5,S_6.

```
sem_t sem_S1, sem_S2, sem_S3, sem_S4, sem_S5, sem_S6, sem_S7
void process_S1(){
        sem_post(sem_S2);   <执行S1
        sem_post(sem_S3);
}
void p_S2(){
    S_wait(S1);   <执行S2
    S_post(S4);
    S_post(S5);
}
void p_S3(){
    S_wait(S1);   <执行S3
    S_post(S6);
}
void p_S4(){
    S_wait(S2);   <执行S4
    S_post(S7);
}
void p_S5(){
    S_wait(S2);   <执行S5
    S_post(S7);
}
Void p_S6(){
    S_wait(S3);   <执行S6
    S_post(S7);
}
void p_S7(){
    S_wait(S4); S_wait(S5);
    S_wait(S6);   <执行S7.
}
int main(){
sem_int(&sem_S1, 0, 0);
    ⋮
sem_int(sem_S7, 0, 0);
process_S1;
    ⋮
}
```

3.

```
Father(){
while(1){
  P(S);
  放水果;
if(橘子) V(So);
else V(Sa);}}

Son(){
while(1){
  P(So);
  取橘子;
  V(S);
  吃橘子;}}
```

3. semaphore empty=5 orange=0 apple=0 mutex=1;

```
Father(){
while(1){
  wait(empty);
  wait(mutex);
  放水果;
  signal(mutex);
  if(橘子) signal(orange);
  else     signal(apple);
}}
```

```
Son(){
while(1){
  wait(orange);
  wait(mutex);
  取橘子;
  signal(mutex);
  signal(empty);
  吃橘子;
}}
```

```
Daughter(){
while(1){
  wait(apple);
  wait(mutex);
  取苹果;
  signal(mutex);
  signal(empty);
  吃苹果;
}}
```

```
main{ cobegin Father(); Son; Daughter(); coend}
```

4. semaphore empty1=1 empty2=1 full1=0 full2=0;

```
PA(){
while(1){
  从磁盘读一个记录;
  P(empty1);
  将记录放入缓冲区1;
  V(full1);
}}
```

```
PB(){
while(1){
  P(full1);
  从缓冲区1中取出记录;
  V(empty1);
  P(empty2);
  将记录放入缓冲区2;
  V(full2);
}}
```

```
PC(){
while(1){
  P(full2);
  从缓冲区2中取出记录;
  V(empty2);
  打印记录;
}}
```

```
main{ cobegin PA(); PB(); PC(); coend;}
```

5. int count=0 semaphore mutex=1, empty=1, full=0, payment=0, receipt=0;

```
customer(){
  wait(mutex);
  if(count>=N){
    signal(mutex);
    离开;}
  else{count++;
    signal(mutex);
    坐沙发;
    wait(empty);
    下沙发,上理发椅;
    wait(mutex);
    count--;
    signal(mutex);
    signal(full);
    理发;付钱;
    signal(payment);
    wait(receipt);
    signal(empty);
    离开;
}}
```

```
Barber(){
    while(1){
        wait(full)
        给客人理发;
        wait(payment);
        收费;
        signal(recipt);
    }}
```

main{cobegin Barber(); Customer(); coend}

第三章：

一. 名词解释

2. 处理机调度：操作系统根据一定的调度算法，从就绪队列中选择一个进程分配给它CPU资源，使其进入运行状态的过程

3. 周转时间：作业从提交到完成所经历的时间。

4. 死锁：两个或多个进程因竞争资源而造成的一种僵局，每个进程都阻等待其他进程手中放弃资源，导致所有进程无法进行。

二. 填空题：

2. 提交 后备 完成

6. 互斥 请求与保持 非抢占 循环等待

五. 问答题

1.

	进程	A	B	C	D	E	平均
FCFS	完成时间	3	9	15	18	20	
	周转时间	3	7	9	12	12	8.6
	带权周转时间	1.00	1.17	2.25	2.40	6.00	2.56
SPF(非抢占)	...	3	9	15	20	11	
	--	3	7	11	14	3	7.6
	---	1.00	1.17	2.75	2.80	1.5	1.84
SPF(抢占)	...	3	15	8	14	10	7.2
	--	3	13	4	14	2	7.2
	---	1.00	2.16	1.00	2.80	1.00	1.59

HRRF	...	3	99	13	20	15	
	...	3	7	9	14	7	6
	...	1.00	1.17	2.25	2.80	3.5	2.14
RR(q=1)	...	4	16	13	14	7	
	...	1.3	2.6	3.2	2.8		
	...						
RR(q=1)	...	4	16	17	20	15	
	...	4	16	13	14	7	10.8
	...	1.33	2.67	3.25	2.8	3.5	2.71

2. A: 20-0-10=10ms
   B: 50-0-10=40ms ∴先A
   C: 50-0-15=35ms

   B: 50-10-10=30ms ∴再C
   C: 50-10-15=25ms

   再第A: 20-20-10=-10ms ∴最B
   B: 50-35-10=5ms
   ∴ A → C → B.

3. (1) Process Allocation (Max-Need) {P0, P3, P4, P1, P2}

Process	Allocation	(Max-Need)
P0	0 0 3 2	
P1	1 0 0 0	
P2	1 3 5 4	
P3	0 3 3 2	
P4	0 0 1 4	

(2) 1. $Request_2(1,2,2,2) \leq Need_2(2,3,5,6)$

2. $Request_2(1,2,2,2) \leq Available(1,6,2,2)$

3. 假设分配，并修改 Available、$Allocation_2$ 和 $Need_2$

$Available = (0,4,0,0)$

$Allocation_2 = (2,5,7,6)$

$Need_2 = (1,1,3,4)$

4. 进行安全性检查：此时对所有进程条件 $Need_i \leq Available(0,4,0,0)$ 都不成立，即 Available 不能满足任何进程的请求，故系统进入不安全状态。

∴ P_2 提出请求 $Request(1,2,2,2)$ 后，系统不能将资源分配给它。

(3) 没有

∵ 此时上述进程并没有申请新的资源，并且得不到资源而进入阻塞状态。

∴ 无法进入死锁